April 4, 1999

Xinetics Deformable Mirror (DM): Initial Data and Characteristics

Contents:

1. Introduction & Physical Characteristics

The goal of the DM project at Goddard has been to verify the DM operation and determine its characteristics for optimizing its control – both in support of the JPL adaptive optical system (implemented in DCATT) and for future Goddard reference.

The Xinetics Goddard DM consists of 349 PMN (lead magnesium niobate) actuators attached to a 2 *mm* thick mirror surface. Clear aperture = 5.85” = 148.6 *mm*. Control aperture = 5.52” (this is the portion of the mirror surface under direct actuator control). The mirror surface consists of a 2 *mm* face-sheet of ultra-low expansion (ULE) glass silver – polished and then bonded onto a circular grid of actuators. The actuator grid is spaced at 7 *mm* increments (center to center) and given the numbering scheme and coordinate system shown in Figure 1. A photograph of the actuator grid is shown in Figure 2 and was taken using a rear view of the DM with the back cover plate removed.

Microsoft Excel - actuator_map.csv

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
1								1	2	3	4	5	6	7								
2						8	9	10	11	12	13	14	15	16	17	18						
3					19	20	21	22	23	24	25	26	27	28	29	30	31					
4				32	33	34	35	36	37	38	39	40	41	42	43	44	45	46				
5			47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63			
6		64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82		
7		83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100	101		
8	102	103	104	105	106	107	108	109	110	111	112	113	114	115	116	117	118	119	120	121	122	
9	123	124	125	126	127	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143	
10	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159	160	161	162	163	164	
11	165	166	167	168	169	170	171	172	173	174	175	176	177	178	179	180	181	182	183	184	185	
12	186	187	188	189	190	191	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	
13	207	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223	224	225	226	227	
14	228	229	230	231	232	233	234	235	236	237	238	239	240	241	242	243	244	245	246	247	248	
15		249	250	251	252	253	254	255	256	257	258	259	260	261	262	263	264	265	266	267		
16		268	269	270	271	272	273	274	275	276	277	278	279	280	281	282	283	284	285	286		
17			287	288	289	290	291	292	293	294	295	296	297	298	299	300	301	302	303			
18				304	305	306	307	308	309	310	311	312	313	314	315	316	317	318				
19					319	320	321	322	323	324	325	326	327	328	329	330	331					
20						332	333	334	335	336	337	338	339	340	341	342						
21								343	344	345	346	347	348	349								
22																						

actuator_map

Figure 1. Xinetics 349 DM Actuator Numbering Scheme

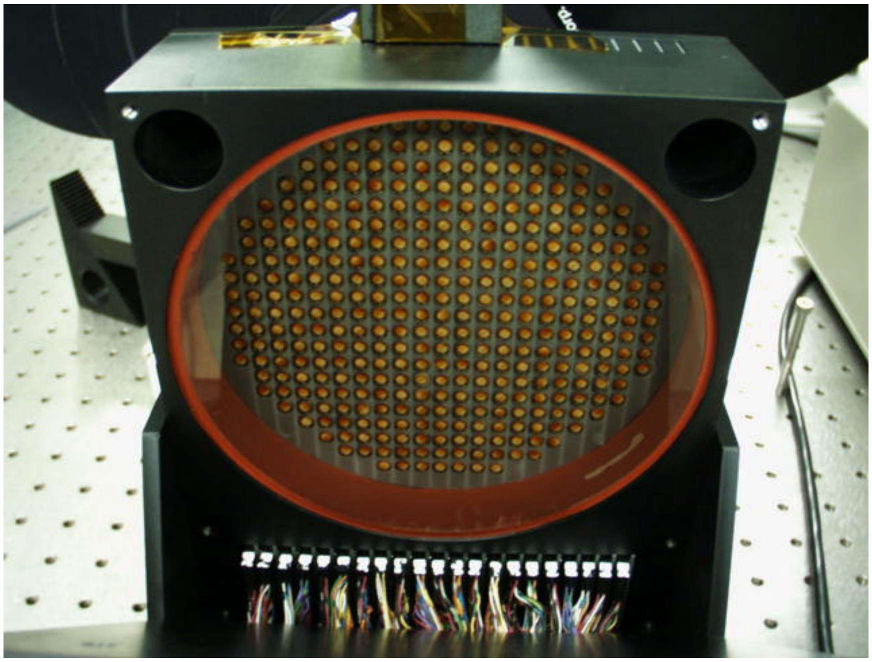


Figure 2 Actuator Grid – DM Rear Cover Plate Removed.

The actuators operate from between $-40V$ to $+100V$ which provides a maximum mechanical stroke of $4mm$. However, the maximum "inter-actuator" displacement between actuators is $2mm$ ($50V$). When maximum inter-actuator displacement is reached, the Zener diode circuitry is activated to prevent further actuator displacement which could potentially damage the mirror surface.

The $-40V$ to $+100V$ range is accessible through 65,536 digital commands (actual values are in the $\pm 32{,}768\,(=2^{15})$ range) sent to the DM electronics via a MAC 950 Quadra (Figure 3 and Figure 4). The -32768 digital commands are mapped to -40V while the +32768 values are mapped to 100V. Using this mapping scheme, the actuator displacements are nearly linear and parabolic otherwise.

To characterize the mirror surface, data was initially taken with a Zygo Mark IV PSI (phase shifting interferometer) - equipped with a 6" beam expander and connected to an HP Apollo 750 running HP-UX. This interferometer was subsequently upgraded to a Zygo model GPI-XP. The computers are networked via DCATTSUN – operating in building 7 (the DM is operating in the CIAF of Building 7).

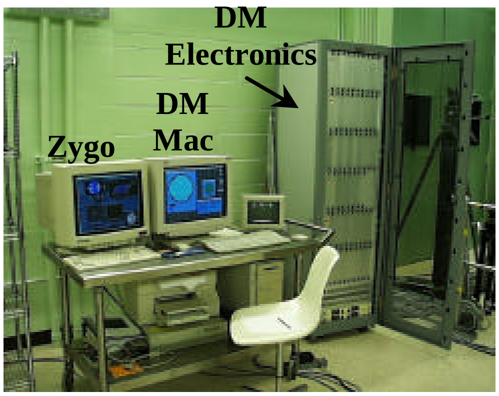


Figure 3. DM Computers and Electronics

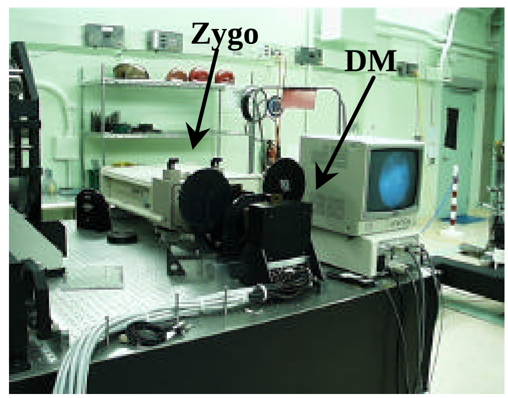


Figure 4. Zygo Interferometer and the Xinetics DM.

2. Power-On State and Preliminary Flattening Results for the DM

On power-up the DM is far from flat due to random inter-actuator displacements that exist on the mm scale. Typically, the power-on mirror surface is characterized by the PV (peak to valley) and *rms* values (1 wave $= 632.8\,nm$):

$$PV = 1.194;\; rms = 0.233\; wave. \tag{1}$$

A Zygo "screen shot" showing the intensity and OPD (optical path difference) maps for this data is given in Figure 5 (note that the *piston* and *tilt* aberrations have been removed by Zygo). For comparison, before powering on the DM, the mirror surface is relatively flat:

$$PV = 0.356;\; rms = 0.046\; wave \tag{2}$$

as illustrated in Figure 6. A summary of the preliminary flattening results is given by

$$PV = 0.277;\; rms = 0.028\; wave \;\approx\; l/36, \tag{3}$$

and shown in Figures 7 and 10 (note: in the case of Figure 7 the intensity map is shown with very little tilt). For comparison of the data in equations (1), (2), and (3), a cross-sections of these OPD maps are displayed in Figures 8, 9, and 10 using identical length scales.

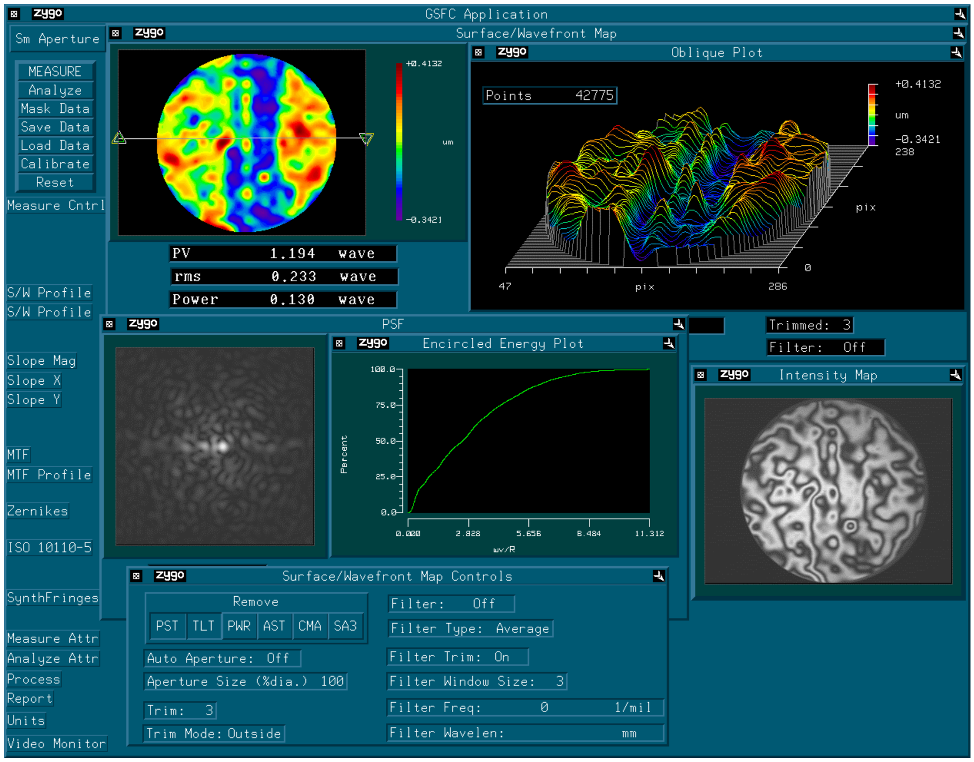


Figure 5. Zygo Data: Power-On State of the DM.

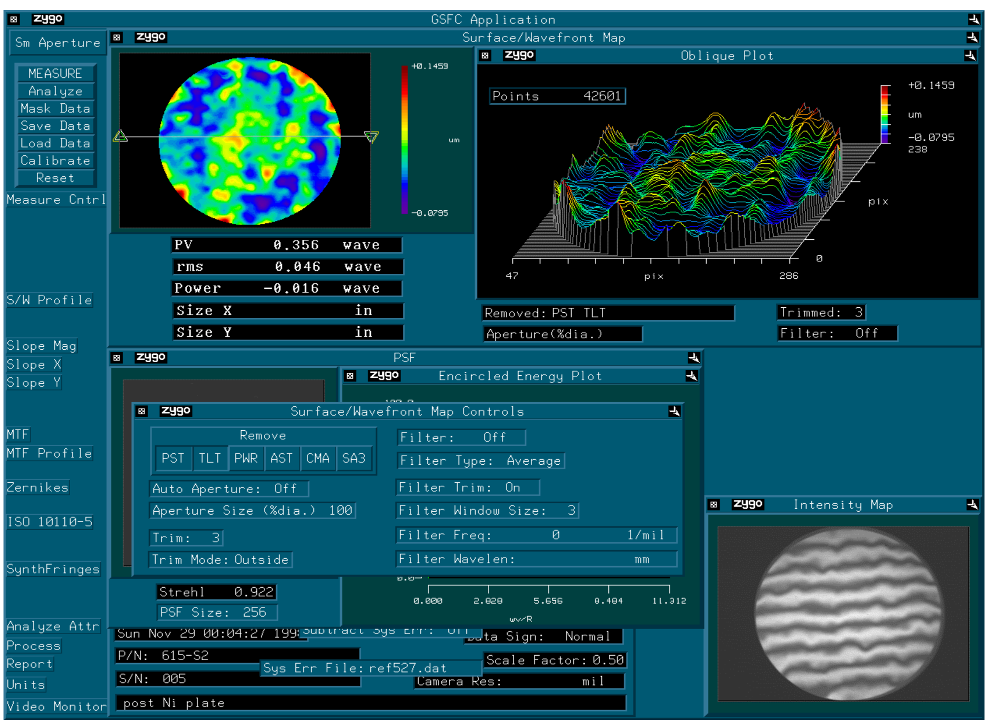


Figure 6. Mirror Surface Before Power-Up.

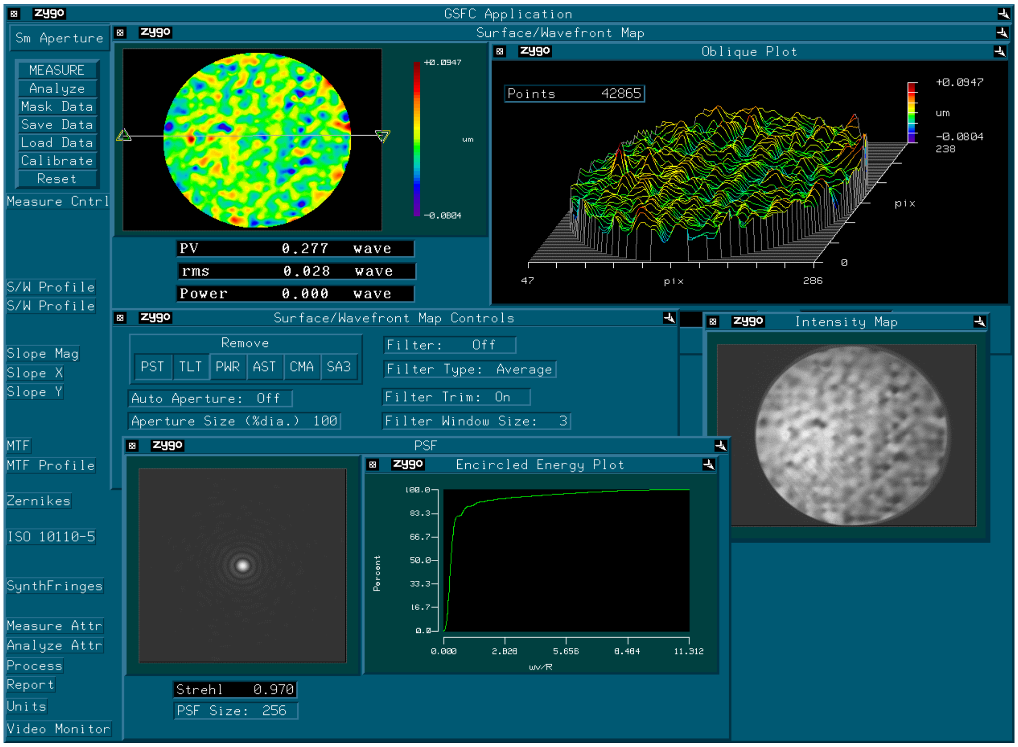


Figure 7. Zygo Data: Preliminary Flattening Results

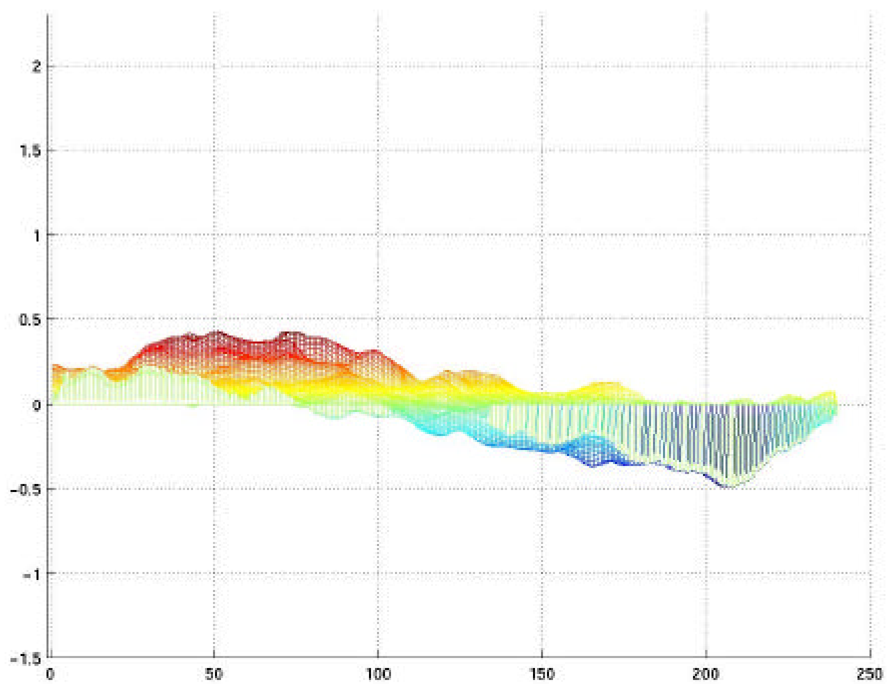


Figure 8. OPD (Optical Path Difference) Cross-Section of the DM in it's No-Power State.

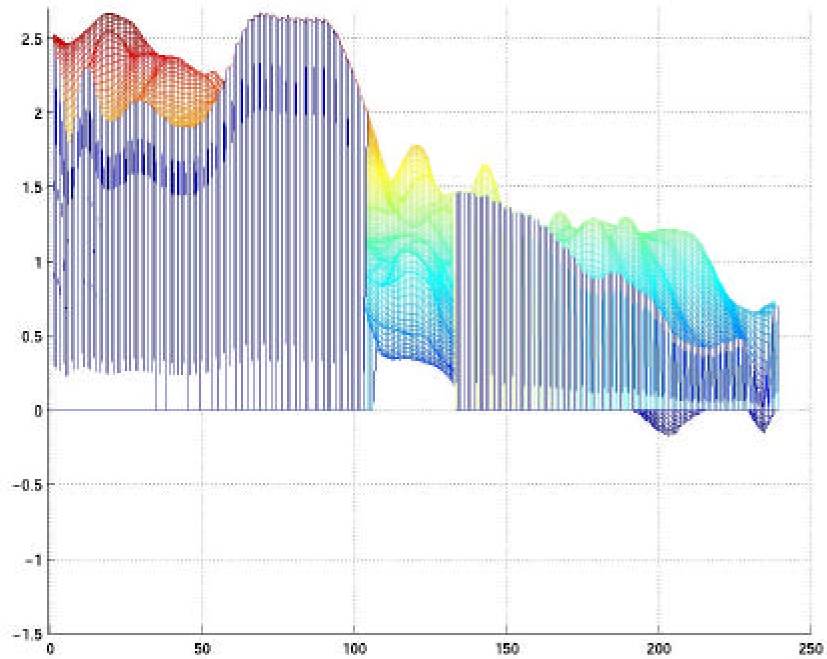


Figure 9. Cross-Section of the DM in its Power-On State.

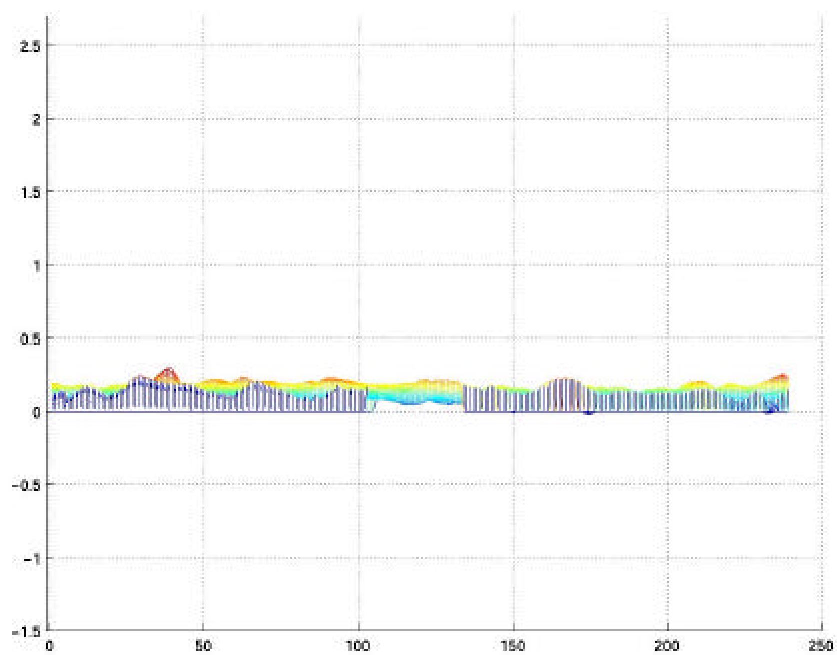


Figure 10. Cross Section after Flattening

3. Influence Function Data

The influence function data file “infl_3c.dat” consists of an initial arbitrarily chosen actuator (# 3C – see Figure 1) and then 3 additional actuators spaced at 10 actuator increments. These actuators form the square grid pattern shown in Figure 11 and are displaced at a +8000 command value relative to the “flattened” mirror surface given by Equation (3) (see also Figures 7 and 10). For reference, the individual actuators are labeled on the intensity map of Figure 11 using the numbering scheme illustrated in Figure 1.

Comparing Figures 1 and 11 shows that the Zygo optical system images an arbitrary image point on the DM surface from *left to right* and then from *top to bottom* to the Zygo display, i.e., creates an inversion of the DM surface. Therefore, this mapping of actuators must be accounted for when sampling the Zygo data at a given actuator position. This aspect is discussed in further detail in the next section describing the MatLab codes for reading the Zygo data.

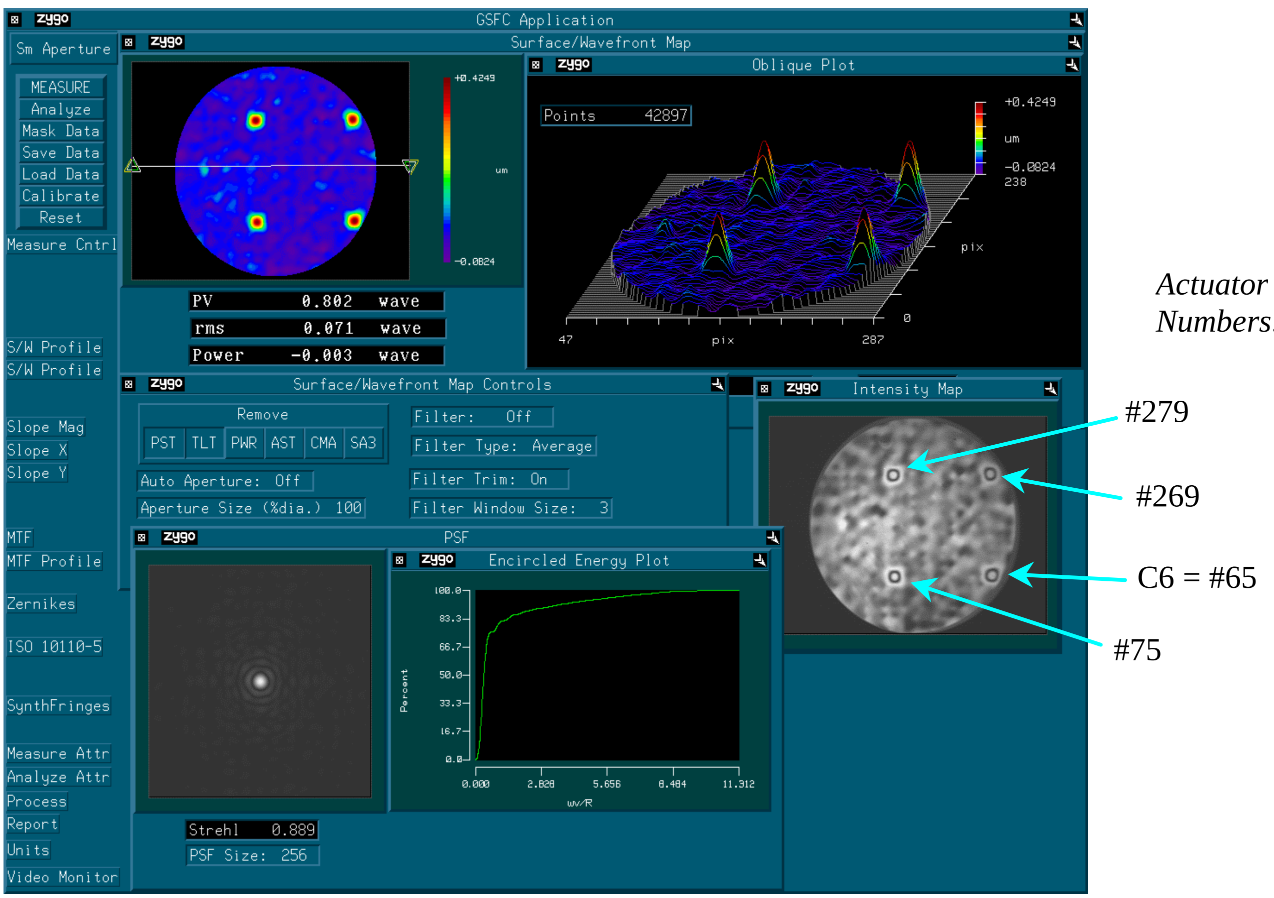


Figure 11. Zygo Screen Shot of “infl_3c.dat”

4. MatLab Codes for Reading the Data

The Zygo influence function data file is stored in the Zygo binary file format. A MatLab code for reading this data, “readzygofile.m” is listed in Appendix 1. This routine calls two additional routines, “readzygoheaderdata.m” and “readzygophasedata.m”, listed in Appendices 2 and 3, respectively. A MatLab code example, “getdata.m”, illustrating how these routines may be used is listed in Appendix 4. The phase data output of “getdata.m” is displayed as a contour map in Figure 12 below.

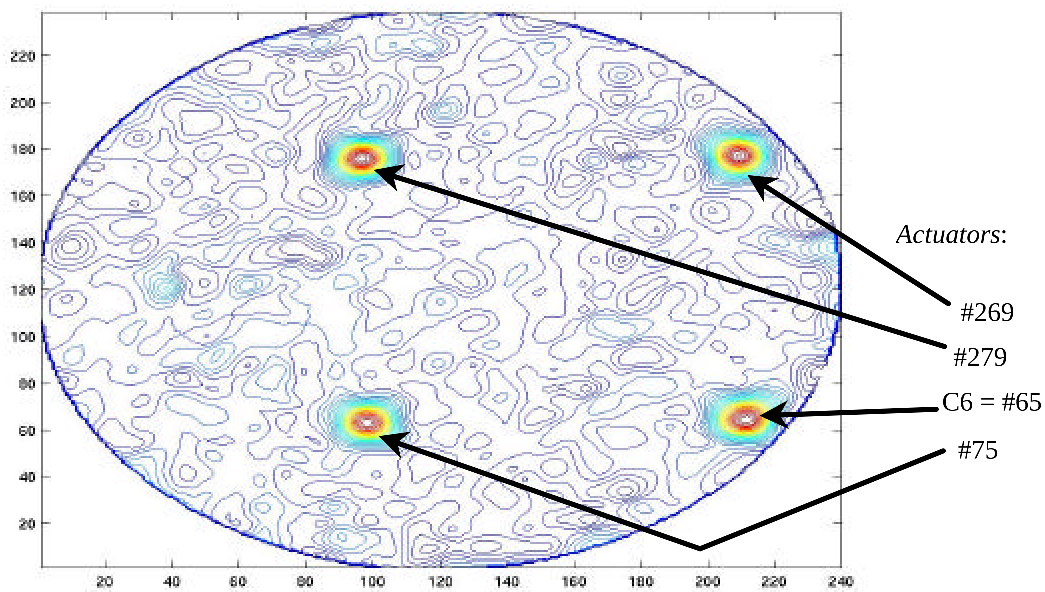


Figure 12. Contour Map of the “infl_3c.dat” with Actuator Numbers Labeled

The phase data (stored in the MatLab variable “phase1” of Appendix 4) is an array whose indices give the *x*-*y* values of the pixel locations at a given data point (see Figure). Typically, these values will range from 0-239 along each axis as shown in the figure above. The *z* value gives the OPD height measured in waves (relative to zero) and is obtained from the array using a command like:

$$z = phase1(y\_value, x\_value). \tag{4}$$

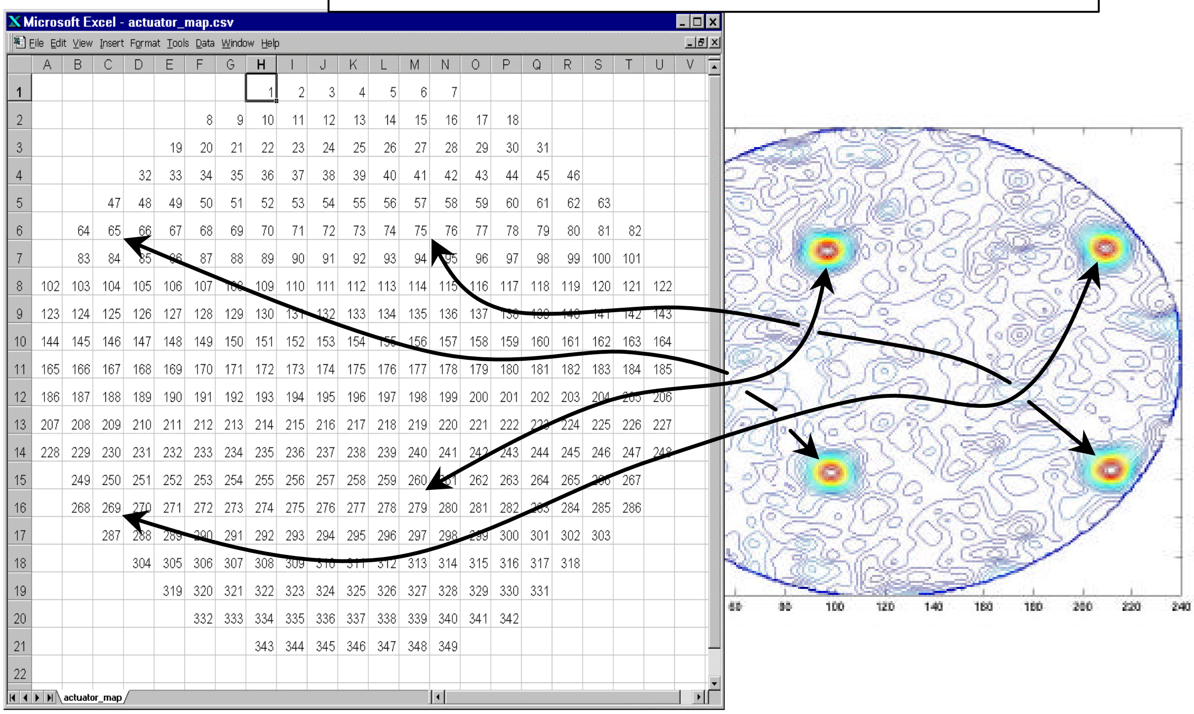
For example (and for JPL reference), the *x*-*y*-*z* values for each of the actuators shown in Figure 12 are given by

$$\begin{array}{cc} \underbrace{z = phase1(64,211) = 1.025}_{actuator\,\#65} & \underbrace{z = phase1(63,98) = 1.459}_{actuator\,\#75} \\ \underbrace{z = phase1(177,209) = 0.978}_{actuator\,\#269} & \underbrace{z = phase1(175,97) = 1.397}_{actuator\,\#279} \end{array} \tag{5}$$

But there is an <u>important detail</u> that should not be overlooked, namely, that to address a given actuator location within the data set, the indices must be reversed to “undo” the transformation given by the segment of code shown in Figure 13 – which orients the MatLab display to match the Zygo display screen (taken from Appendix 4). This transformation is illustrated graphically in Figure 13 for the actuators displayed in the “infl_3c.dat” data set.

```
% Flip the phase data to match the Zygo display:

sy = header.PhaseHeight:-1:1;
phase1 = phase([sy],:);
```



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
1								1	2	3	4	5	6	7								
2						8	9	10	11	12	13	14	15	16	17	18						
3					19	20	21	22	23	24	25	26	27	28	29	30	31					
4				32	33	34	35	36	37	38	39	40	41	42	43	44	45	46				
5			47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63			
6		64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82		
7		83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100	101		
8	102	103	104	105	106	107	108	109	110	111	112	113	114	115	116	117	118	119	120	121	122	
9	123	124	125	126	127	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143	
10	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159	160	161	162	163	164	
11	165	166	167	168	169	170	171	172	173	174	175	176	177	178	179	180	181	182	183	184	185	
12	186	187	188	189	190	191	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	
13	207	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223	224	225	226	227	
14	228	229	230	231	232	233	234	235	236	237	238	239	240	241	242	243	244	245	246	247	248	
15		249	250	251	252	253	254	255	256	257	258	259	260	261	262	263	264	265	266	267		
16		268	269	270	271	272	273	274	275	276	277	278	279	280	281	282	283	284	285	286		
17			287	288	289	290	291	292	293	294	295	296	297	298	299	300	301	302	303			
18				304	305	306	307	308	309	310	311	312	313	314	315	316	317	318				
19					319	320	321	322	323	324	325	326	327	328	329	330	331					
20						332	333	334	335	336	337	338	339	340	341	342						
21								343	344	345	346	347	348	349								
22																						

Figure 13. “Transformation” of the DM Surface Through the Zygo Optical System

5. Appendix 1 – MatLab Routine: readzygofile.m

```
function [header, phase, success] = readzygofile2(filename)
%
% ReadZygoFile reads all the data from the specified Zygo file.
%
% Input:
%   filename = the name of the Zygo file to read.
%
% Output:
%   header = a structure containing selected contents from the file header.
%   intens = an array containing the intensity data from the file.
%   phase = an array containing the phase data from the file.
%   success = a scalar indicating success (1) or failure (0) of the function.
%
% Versions:
%09/14/98 Ladd Wheeler Original version
Modified by Bruce Dean and Anand Sivaramakrishnan

success = 1;

% Open the file for read-only binary access with big-endian format.
file = -1;
[file, message] = fopen (filename, 'r', 'ieee-be');
if file == -1
    success = 0;
    error (message);
end

% Now read the header data.
[header, success] = readzygoheaderdata (file);
if success == 0
    error ('Failure in reading Zygo header data');
end

% Now read the phase data.

[phase, success] = readzygophasedata (file, header);

if success == 0
    error ('Failure in reading Zygo phase data');
end

% Close the file.
fclose (file);
```

6. Appendix 2 – MatLab Routine: readzygoheaderdata.m

```
function [header, success] = ReadZygoHeaderData (file)
%
% ReadZygoHeaderData reads selected data from the specified Zygo file.
%
% Input:
%   file = the file id of the Zygo file to read.
%
% Output:
%   header = a structure containing selected contents from the file header.
%   success = a scalar indicating success (1) or failure (0) of the function.
%
% Versions:
%   09/15/98 Ladd Wheeler  Original version

success = 1;

% Read Magic Number
status = fseek (file, 0, 'bof');
if  status ~= 0
    success = 0;
    error ('Failure to seek to 0 in file.');
end
[magic_number] = fread (file, 1, 'int32');
if  magic_number ~= -2011495569
    success = 0;
    error ('Magic number in file is not valid.');
end

% Read Header Format
header_format = fread (file, 1, 'int16');
if  header_format ~= 1
    success = 0;
    error ('Header format value in file is not 1.');
end

% Read Header Size
header_size = fread (file, 1, 'int32');
if  header_size ~= 834
    success = 0;
    error ('Header size value in file is not 834.');
end

% Read Intensity Description Data
status = fseek (file, 48, 'bof');
if  status ~= 0
    success = 0;
    error ('Failure to seek to Intensity Origin values in file.');
end
header.IntensOriginX = fread (file, 1, 'int16');
header.IntensOriginY = fread (file, 1, 'int16');

header.IntensWidth = fread (file, 1, 'int16');
header.IntensHeight = fread (file, 1, 'int16');

header.NBuckets = fread (file, 1, 'int16');
header.IntensRange = fread (file, 1, 'uint16');
header.IntensBytes = fread (file, 1, 'int32');

% Read Phase Description Data
header.PhaseOriginX = fread (file, 1, 'int16');
header.PhaseOriginY = fread (file, 1, 'int16');

header.PhaseWidth = fread (file, 1, 'int16');
header.PhaseHeight = fread (file, 1, 'int16');

header.PhaseBytes = fread (file, 1, 'int32');
```

```
% Read Time Stamp and Comment
header.TimeStamp = fread (file, 1, 'int32');
header.Comment = fread (file, 82, 'char');

% Read Other
header.Source = fread (file, 1, 'int16');
% **************************************************
%dummy = fread (file, 1, 'char');
% **************************************************
header.IntfScaleFactor = fread (file, 1, 'float32');
header.Wavelengthin = fread (file, 1, 'float32');
header.NumericAperature = fread (file, 1, 'float32');
header.ObliquityFactor = fread (file, 1, 'float32');
skip = fread (file, 1, 'float32');
header.CameraRes = fread (file, 1, 'float32');

% Read Next Clumps of Descriptives
status = fseek (file, 218, 'bof');
if  status ~= 0
    success = 0;
    error ('Failure to seek to PhaseRes in file.');
end
header.PhaseRes = fread (file, 1, 'int16');
header.MinimumAreaSize = fread (file, 1, 'int32');
header.DisconAction = fread (file, 1, 'int16');
header.DisconFilter = fread (file, 1, 'float32');
header.ConnectionOrder = fread (file, 1, 'int16');
header.DataSign = fread (file, 1, 'int16');
header.CameraWidth = fread (file, 1, 'int16');
header.CameraHeight = fread (file, 1, 'int16');

status = fseek (file, 300, 'bof');
if  status ~= 0
    success = 0;
    error ('Failure to seek to PhaseAvgs in file.');
end
header.PhaseAvgs = fread (file, 1, 'int16');
header.SubtractSysErr = fread (file, 1, 'int16');

status = fseek (file, 364, 'bof');
if  status ~= 0
    success = 0;
    error ('Failure to seek to RemoveTiltBias in file.');
end
header.RemoveTiltBias = fread (file, 1, 'int16');
header.RemoveFringes = fread (file, 1, 'int16');
header.MaxAreaSize = fread (file, 1, 'int32');
skip = fread (file, 1, 'int16');
header.PreConnectFilter = fread (file, 1, 'float32');
```

7. Appendix 3 – MatLab Routine: readzygophasedata.m

```
function [phase, success] = readrygorhaserata (file, header)
%
% ReadZygoPhaseData reads Phase data from the specified Zygo file.
%
% Input:
%   file = the file id of the Zygo file to read.
%   header = a structure containing selected contents from the file header.
%
% Output:
%   phase = an array containing the phase data.
%   success = a scalar indicating success (1) or failure (0) of the function.
%
% Versions:
%   09/15/98 Ladd Wheeler  Original version.

success = 1;

% Check if any Phase Data is in file
if  header.PhaseBytes == 0
    error ('There is no Phase data in the file.');
end

% Seek to the beginning of the Intensity Data
offset = 834 + header.IntensBytes;
status = fseek (file, offset, 'bof');
if  status ~= 0
    success = 0;
    error ('Failure to seek to the Phase Data in the file.');
end

% Preallocate the phase array to speed up the execution.
phase = zeros (header.PhaseHeight,  header.PhaseWidth);

% Read the Phase Data
%   According to Zygo documentation, the data is written in "row-major order".
%   This unit assumes that the scan starts at the upper left (row 1, col 1)
%   and moves left-to-right along EACH row.

for row = 1:header.PhaseHeight
    for col = 1:header.PhaseWidth
        phase(row,col) = fread (file, 1, 'int32');
    end
end
```

8. Appendix 4 – MatLab Routine: getdata.m

```
% This is "getdata.m" - a sample MatLab code for reading Zygo data

% This is the function call to Ladd's routine for reading in the
% Zygo phase data:

Portions of this code were contributed by Anand Sivaramakrishnan.

success = 1;
[header, phase, success] = readzygofile2('/home/aodm/src/data/infl_3c10.dat');

% Legal phase data range is (inclusive of these values):

MINPHASE = -2097152;
MAXPHASE =  2097152;

% Initialize two variables that will be used to eliminate
% bad data points.

phase_setfunc = zeros (header.PhaseWidth, header.PhaseHeight);
phase_setfunc = phase_setfunc  + 1;

% Eliminate the bad data points (Anand Sivaramakrishnan)):

for row = 1:header.PhaseHeight
   for col = 1:header.PhaseWidth
   if ((phase(row, col) < MINPHASE) | ...
      (phase(row, col) > MAXPHASE))
       phase(row, col) = 0;
       phase_setfunc(row, col) = 0;
    end
   end
end

% Convert the binary data to wave units:

phase = phase * header.IntfScaleFactor * header.ObliquityFactor / 4096;

% Flip the phase data to match to match the Zygo display:

sy = header.PhaseHeight:-1:1;
phase1 = phase([sy],:);

figure(1);
contour(phase1,20);
```